Migrating from UIKit to SwiftUI efficiently

```swift
final class MyViewController: UITableViewController {
    private var data = [String]()

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "cell")
        let d = data[indexPath.row]
        cell?.textLabel?.text = d

        return cell ?? UITableViewCell()
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return data.count
    }
}
```

↓

```swift
struct MyView: View {
    private var data = [String]()

    var body: some View {
        List(data, id: \.self) { d in
            Text(d)
        }
    }
}
```

Self Intro

— Name: Joshua Kaplan
— Work: minne @ GMO Pepabo
— Interests: 🤖CI/CD, 📦frameworks, and more
— Hobbies: 🥯bread, 📚history, and 🏃running

Intro

1. Why move to SwiftUI?
2. Why not move to SwiftUI?
3. Modernize Swift Usage
4. Modernize UIKit usage
5. Plan and prototype
6. Two approaches
7. Tips

Why move to SwiftUI

— Do more with less code (for most things)
— Easier to implement
— The future of GUI development

Why not move to SwiftUI

— Stability
— iOS 12 and less compatibility
— Low-level or high performance needs
— Mixing can be difficult and painful without planning
— How urgent?

Modernize Swift usage

— Migrate from Objective-C!
— Use latest Swift version
— Use Swiftier conventions
— Use all the latest features
— Get familiar with FRP frameworks

Modernize UIKit usage

— Use auto layout
— Support safe area
— Components
— Thin or no storyboards
— Dynamic type and dark mode
— Use declarative UIKit APIs

Plan and prototype

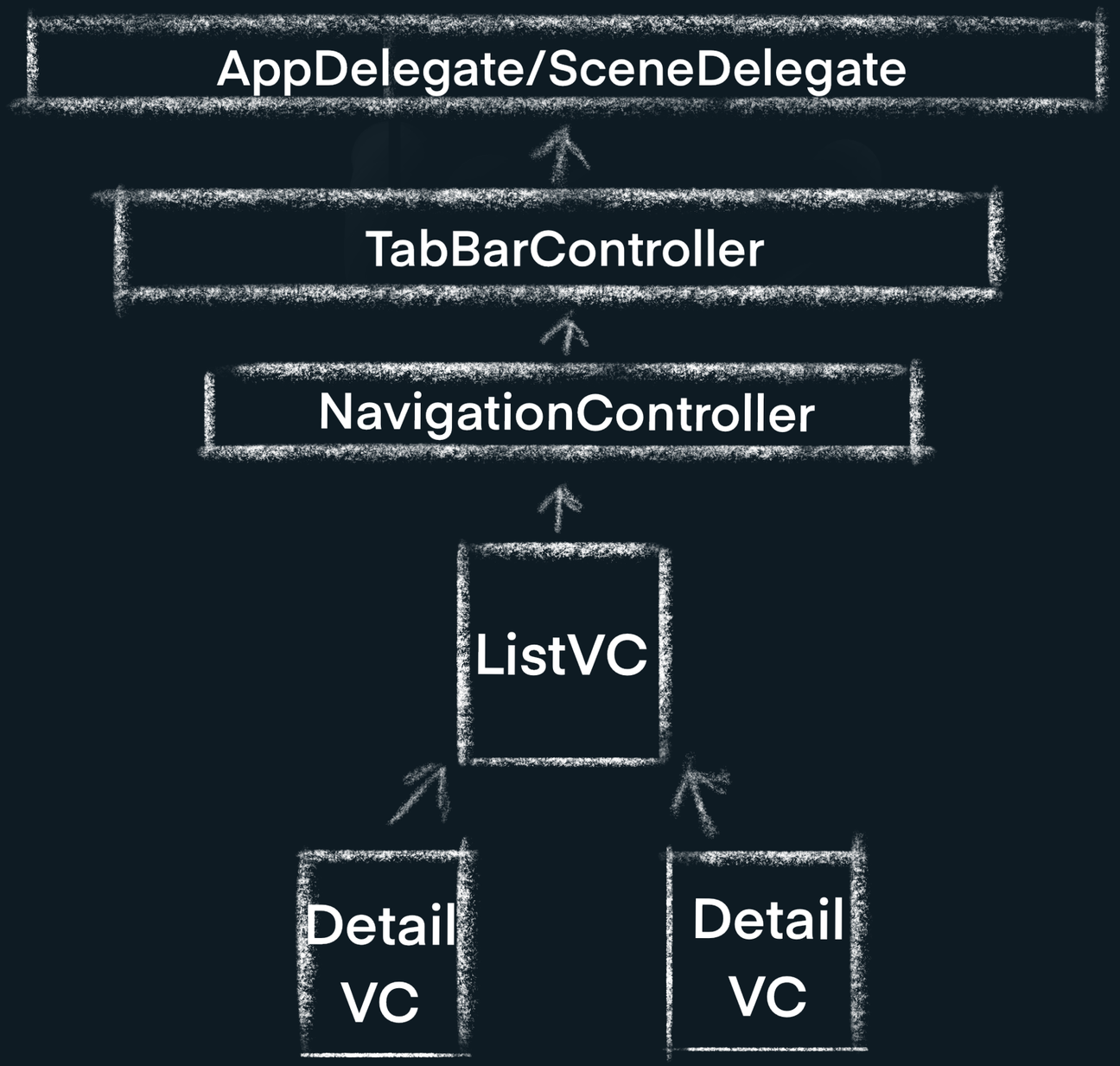
Prototype

— Make a prototype
— Identify screens/components not suited to SwiftUI
— Change the design
— Make an iOS 13-only feature

Architecture

— Redux
— The Composable Architecture (TCA)
— MVVM

Two approaches


AppDelegate/SceneDelegate
TabBarController
NavigationController
ListVC
Detail VC
Detail VC

Tips

— Don't mix too much
— Start with easier screens
— Don’t hurry
— Study SwiftUI and Combine in advance

Conclusion

Thank you

Reference

Prototype

— Shop.app

Combine

Docs

— Official Documentation
— RxSwift to Combine Cheatsheet
— Combineフレームワークまとめ

Code

— CombineSwiftPlayground
— Combine-MVVM
— OpenCombine

Video

— Introducing Combine

— Combine in Practice

Books

— Using Combine
— Practical Combine
— Understanding Combine
— Combine: Asynchronous Programming with Swift

SwiftUI

— Official Documentation

Books

— Thinking in SwiftUI
— SwiftUI by Tutorials